Helsinki University of Technology
CFD-group/ Laboratory of Applied Thermodynamics

MEMO No CFD/TERMO-7-96 DATE: April 17, 1996

TITLE

Transportation of FINFLO to CRAY T3D

AUTHOR(S)
Patrik Rautaheimo

ABSTRACT

Explanation of the implementation of FINFLO to the CRAY T3D and also report the benchmark results.

MAIN RESULT

Benchmark runs in CRAY T3D

PAGES
5

KEY WORDS
Massively parallel computing, CRAY T3D, MPP, CFD, MPI

APPROVED BY

Timo Siikonen April 17, 1996

1 General set up

In this work the CRAY T3D computer is used for benchmark the FINFLO CFD code. Computer is located in Eagan, USA. One purpose of this work is made as easy as possible to transport FINFLO to coming CRAY T3E in CSC, Finland. Also the knowledge of the performance of FINFLO software in massively parallel (MPP) machines is main goal. Most of the documents, that was used in this document, where found from Edinburgh Parallel Computing Centres www-server, **http://www.epcc.ed.ac.uk/t3d/index.html**. Separate reference list is not made.

2 Compiling the program

Compiling was done in CRAY J90. Only minor changes were needed to compile program. For monitoring the `SECOND` subroutine was not supported and `IRTC` was used. `IRTC` returns the time in clock cycles. It can be multiplied by $6.667 \cdot 10^{-9}$ to get the time in seconds.

It is not possible to pass FORTRAN `CHARACTER` arrays and strings as the choice buffer argument to MPI calls. The following FORTRAN code fragment illustrates this workaround, using equivalent `INTEGER` array

```
      CHARACTER*80 BCAPU(1000)
      EQUIVALENCE(BCAPU(1),IBOGUS)
      INTEGER IBOGUS(10*1000)
C ... CRAY do not support CHARACTER send in MPI.

-----------------------------------

C      CALL MPI_SEND(BCAPU(1),80*NPCSL,MPI_CHARACTER,NPRO-1,500+NPRO,
C     +     MPI_COMM_WORLD,IERR)
      CALL MPI_SEND(IBOGUS(1),80*NPCSL,MPI_CHARACTER,NPRO-1,500+NPRO,
     +     MPI_COMM_WORLD,IERR)
```

Same must be done for `MPI_RECV` command.

Makefile was the following:

```
SHELL = /bin/sh
F77 = /mpp/bin/cf77

WORKS= res3c.o imps3.o base3c.o turb3.o tgas.o ugas.o outp3.o  state.o\
gluebm.o rotdia.o bound.o reynol.o fscal.o scimp.o chim.o messp.o
DYNA = ns3c.o main.o $(WORKS)

SOURCES = res3c.f imps3.f base3c.f turb3.f outp3.f  state.f \
gluebm.f rotdia.f fscal.f scimp.f bound.f reynol.f chim.f messp.f

FFLAGS = -dp -Oscalar3

install: all
all: finflo
clean:
        rm -f *.l *~
veryclean: clean
        rm -f *.o  finflo

finflo: $(DYNA)
        $(F77) -I/usr/include/mpp -C cray-t3d $(DYNA) -o finflo -lmpi
```

```
ns3c.o main.o  : NS3CH.C WORK3.C NS3CO.D NS3CO.CC NS3CO.PAC
        $(F77) -I/usr/include/mpp -C cray-t3d $(FFLAGS) -c $*.f

$(WORKS) : WORK3.C NS3CH.C
        $(F77) -I/usr/include/mpp -C cray-t3d $(FFLAGS) -c $*.f
```

The flags `-dp` and `-Oscalar3` make `DOUBLE PRECISION` as `REAL` and aggressive scalar optimization. Flag `-C cray-t3d` tells the CRAY J90 to make binary for the T3D and `-lmpi` link the `MPI`-libraries.

For debugging the program the number of processors must be specified for a compiler. Also `-g` flag can be defined that enables variable in program `totalview`.

3 I/O on the T3D

Grids were transfer in ASC mode by using `ftp`. Formal to binal transformation was performed in the J90 by using programs `formal.f` and `binal.f`.

The J90 uses a different floating point format to the T3D. The FORTRAN IO library will perform the necessary conversion if requested. This is done by using the `assign` command before running the T3D. Command is

```
CRAY /ptmp/n6719> assign -N cray /ptmp/n6719/DELTA2B.GRID
```

This request remains in force until you log out or until removed or replaced by a new call to assign. Other possibility could have been running the `binal.f` program directly in the T3D. This was not tested.

During test runs some inconsistency arise with this command. It was noticed that the command must be ran in same directory as the program is ran. Also the command was changed in other directory during calculations. At the end of these calculation the command could be found from: `/opt/ctl/craylibs/craylibs/bin/assign`.

In present case problems arise with open the files. It seems that in an account used, there was open files where limited to the 50 (writer do not where to find this indication). However in present account limit was set to the 1024. This can be checked by a command:

```
CRAY /finflo/Src> udbsee | grep pfd
        pfdlimit[b]     :1024:
        pfdlimit[i]     :1024:
CRAY /finflo/Src>
```

The limit can be changed to the maximum by using command:

```
limit -v -f 1024 -p0
```

This command do not work in t-shell. The work was done by batch job that is written in bourne shell.

4 Running the program

Program can be automatically run in in T3D from J90. User must only specify the number of processors needed in the run:

```
CRAY /tmp/n6719> time ~/finflo/Src/finflo -npes 1
```

Program could also been compiled for some particular number of processors and then no `-npes` option is needed.

After some test calculation it was realized that it is better made runs by using batch job. This was because of this tricky I/O. Example of one batch file is:

```
#!/bin/sh
# @$ -eo
#QSUB -l mpp_p=4,mpp_t=899 # mpp_p number of processors
#QSUB -lM 4Mw    #amount of requested memory
#QSUB -lm 4Mw    #amount of requested memory
#QSUB -lT 899    #amount of requested time
#QSUB -lt 899    #amount of requested time
cd /tmp/n6719
/opt/ctl/craylibs/craylibs/bin/assign -N cray /ptmp/n6719/DELTA4B.GRID
limit -v -f 1024 -p0
ja
/rain/u2/n6719/finflo/Src/finflo -npes 4
ja -sclf
```

5 Debugging

If the environment variable `TRACEBK` is set (`setenv TRACEBK 0`) and program has an exception, it will generate an `mppcore` file. Totalview is an X-Based debugger that interprets this core file. If the program was compiled with the `-g` or `-G` debug options, source code and global variables will be available to Totalview. In this case the process window will display the source and point to the line where the exception occurred. Code must be compiled with the process number `-X` flag. To run this, type:

```
CRAY /tmp/n6719> totalview /ptmp/n6719/src2/finflo mppcore &
```

where `/ptmp/n6719/src2/finflo` can be any program.

6 Results

The CRAY T3D results were not able to do in an empty machine and the results varies during time. The average was taken. Also test with SGI workstations were performed during there was other communication in low-speed Ethernet.

Linear scaling

As a first test case, Delta wing was calculated with different size of the grids. Ten iteration cycles were performed during each run. Main objective was to keep one process job equal. Every process simulates block size of $32 \times 32 \times 32$. The computational domain was split into $1 - 64$ different blocks and each block was calculated in a different process. Thus the coarsest grid has 32 768 and the densest grid 2 097 152 grid points. The performance was measured trough `IRTC` that counts cycles of the processor. Absolute time was get by dividing number by $150 \cdot 10^6$. Each process counts its own computing time. Since it was not possible to obtain T3D results in an empty machine, the results vary during time. The minimum of these time is presented. Time of the pre and post processing of the program was not included in the time. Generally this took from 38 seconds (with 1 PEs) to the 97 seconds (with 64 PEs). Computation times can be seen in Table 1. Total time means wall time for program executing. This includes the pre and post processing of the program. In table CPU means wall time spend in iterations cycles, mus is CPU/CYCLE/CELL in micro seconds. Variable t_{com}/t_{tot} means ratio between communication and calculation. At the end of the calculations it was noticed that there was extra communications because of interruption possibility of the program. This was removed in last case. This can be seen as a increase of communication time and decrease of total time and the calculation time. Variable η is efficiency. Speed up can be seen in Fig. 1 with the theoretical maximum speed-up and also

Table. 1: Performance of the parallelization with linear scaling.

Nodes	Total time	CPU	mus	N. of cells	t_{com}/t_{tot}	η T3D	η SGI
1	576.52	538.02	820.96	32 768	0.001	1.000	1.0
2	600.04	542.17	830.80	65 536	0.004	0.992	0.950
4	599.12	549.76	838.98	131 072	0.004	0.978	0.850
8	603.11	549.96	839.22	262 144	0.015	0.978	0.904
16	605.15	546.43	833.84	524 288	0.016	0.985	0.859
32	630.29	558.41	852.13	1 048 576	0.015	0.963	N/A
64	696.13	589.34	900.78	2 097 152	0.036	0.913	N/A

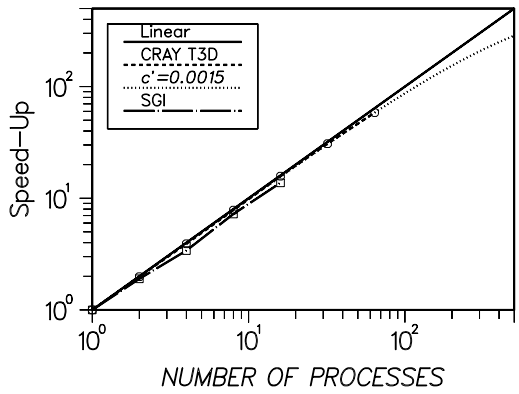

Fig. 1: Performance of the parallelization.

with SGIs cluster of workstations with a standard low speed ethernet. Also estimated speed up for higher number of processes is estimated through Amdahl's law

$$\text{Speed up} = \frac{N}{1 + c^{'} N} \tag{1}$$

where N is number of processes and $c^{'}$ time spend spend in communication per time spend in single processor.

One iteration takes 1.0068 CPU seconds in CRAY C90 (CSC). The C90 was not fully empty and only 68.4% of processor time was used. Speed was with 100 cycles run 303 Mflops. This means that speed of one processor in T3D is about 11 Mflops. Also the performance of the 64 nodes case was $64 \cdot 0.914 \cdot 11$ Mflops = 643 Mflops.

Blocking

Other way of testing parallelization is divide big problem to small ones. Here the case was same Delta wing with grid size $128 \times 64 \times 64$. Problem arose with memory of the one processor of Cray T3D. Duo to memory limitation only 16, 32 and 64 node cases were able to make.

Table. 2: Performance of the parallelization with blocking.

Nodes	Total time	CPU	mus	N. of cells	η T3D
16	597.38	547.04	831.29	32 768	1.000
32	345.92	290.34	881.14	16 384	0.942
64	304.55	213.49	1278.32	8 192	0.641

As it can been seen from Table 2 the scaling is not so good with blocking. This is due to fact that FINFLO calculates "extra" ghost cells for every block. Smaller the block

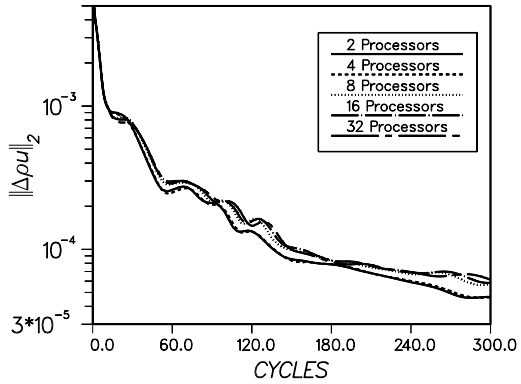

Fig. 2: L_2-norm of x-momentum residual.

is, relatively larger is number of ghost cells. Also the face of the block were not equal sizes. Also the interruption possibility was included with these calculation that reduced the performance.

Splitting a computational domain in smaller parts will reduce the performance of the implicid sweep. However in this case the effect is minor. Iteration history of L_2-norm of x-momentum can be seen in Fig. refkuva5.

7 Discussion

Eagans CRAY T3D was all the time loaded with other works. Performance varies somewhat during runs. The average was taken, but because of CPU time limit, no massively calculations were made.

For case of current configuration of coming CRAY T3E (192 nodes), the FINFLOs message passing seems to perform excellent. But for future the message passing must be optimized. From Fig. 1 the estimated speed up with 500 nodes is only 280 that is not acceptable. From this work it seems that FINFLO should be made runing faster. This can be done through optimizing communication subroutines in FINFLO.

One node performance of CRAY T3D was poor. FINFLO runs speed of 26 Mflops in SGI Indigo^2 (200MHz R4400 with 1Mb secondary cache), about 62 Mflops in SGI Power Challenger (75MHz R8000), 303 Mflops in CRAY C94 and only 11 Mflops in T3D. Poor performance of T3D is because of it has no secondary cache. Time cycle in T3E is double of T3D (300MHz and 150MHz). T3E should have better port for data in processors and thus the performance of one node should be better. However T3Es processor should perform toughly 10 times better that T3D to be satisfactory up-to-date processor. It was good that SGI bought Cray.